OIPE JC112 AUG 22 2005 RADEMARK OFFICE

AF/ [illegible]

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	§	Group Art Unit: 2133
	§	
James M. Byrd	§	Examiner: Gandhi, Dipakkumar B.
	§	
	§	Atty. Dkt. No.: 5181-94400
	§	P6301
Serial No. 10/037,361	§	
Filed: October 29, 2001	§	
For: SYSTEM AND METHOD FOR VERIFYING ERROR DETECTION/CORRECTION LOGIC	§	

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

August 19, 2005 [illegible]
Date Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal mailed June 21, 2005, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

08/23/2005 MAHMED1 00000026 501505 10037361
01 FC:1402 500.00 DA

I. REAL PARTY IN INTEREST

As evidenced by the assignment recorded at Reel/Frame 012455/0469, the subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-35 are pending and finally rejected. The rejection of claims 1-35 is being appealed. A copy of claims 1-35 is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

Claims 1 and 13 were amended subsequent to the final rejection to correct minor typographical errors (see Amendment filed May 25, 2005). In the Advisory Action of June 17, 2005, the Examiner indicated that these amendments would be entered. The copy of the claims in the Claims Appendix includes these amendments as having been entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Data storage or transmission systems commonly implement various types of error detection and/or correction schemes in order to ensure data integrity in the face of transient error conditions, such as noise or crosstalk, or permanent error conditions, such as component failures. *See, e.g.*, specification, p. 1, lines 10-26. Error detection/correction logic may itself be susceptible to such transient or permanent conditions. Since proper operation of a device that implements error detection/correction logic may depend on such logic operating correctly, it may be desirable to test such logic, e.g., during manufacturing, initialization or operation. However, typical tests for error detection/correction logic tend to input a large number of test values into the logic being tested, and as a result, the verification may take a significant amount of time to perform. Furthermore, even though large numbers of test values may be used, these test values may not be selected to thoroughly test each component of the error detection/correction logic, and thus the tests may not provide a complete verification of the error detection/correction logic. *See, e.g.*, specification, p. 2, line 24 – p. 3, line 2.

Independent claim 1 is directed to a method of testing error correction/detection logic. The method includes creating an initial data bit combination having n bits, wherein each data bit in the initial data bit combination has a same logical value as each other data bit in the initial data bit combination; shifting a first bit having a different logical value than the same logical value across the initial data bit combination, wherein each time the first bit is shifted, one of n data bit combinations is generated; providing each of the n data bit combinations to the error detection/correction logic; in response to said providing, the error detection/correction logic generating a set of check bits for each of the n data bit combinations; comparing the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations; and dependent on an outcome of said comparing, generating an indication of whether the error detection/correction logic correctly generated the set of check bits. One embodiment of such a method is illustrated in FIG. 5. As shown at 501 and described in the specification at p. 18, lines 9-28, a test data bit generator (e.g., test data bit generator 202 of FIG. 4A) may create a set of n data bit combinations by creating an initial combination of bits all having the same value and shifting or "walking" a bit having a different value across the initial combination. The resulting n data bit combinations may then be provided as test inputs to test error detection/correction logic, such as check bit generator 102 of FIG. 4A. As shown at 503-509 and described in the specification at p.18, line 30 – p. 19, line 16, check bit generator 102 may then generate a set of check bits, such as parity and/or ECC bits, from the n data bit combinations. The resulting check bits may be compared against a known correct set of check bits, and an indication of whether the check bits were correctly generated may be generated in response to the comparison. For example, as shown in FIG. 4A and described in the specification at p. 14, line 25 – p. 15, line 7, a tester 200 may maintain correct check bit values in a table 208, and comparison logic 206 may compare correct check bit values against the values generated by a device under test (e.g., encoder 100) to determine a verification signal 220 indicating whether the device under test successfully processed a particular one of the n data bit combinations.

Independent claim 13 recites a computer readable medium comprising program instructions computer-executable to perform actions similar to the limitations recited in claim 1. For example, as shown in FIG. 3 and described in the specification at p. 12, lines 9-23, a computer system 3000 may store instructions within system memory 3003 that are executable by a processor 3001 to perform in software the testing operations upon error correction/detection logic 100 that were described above with respect to tester 200.

Independent claim 24 is directed to a test for testing error correction/detection logic, the tester comprising test check bit generating means for creating a set of test data bit combinations and providing the set of test data bit combinations to a check bit generator comprised in the error correction/detection logic, wherein the set of test data bit combinations comprises n n-bit data bit combinations, wherein each possible logical value of each data bit is present in at least one of the n n-bit data bit combinations in the set of test data bit combinations. The tester further includes comparison means for comparing check bits output by the error correction/detection logic for each of the n n-bit data bit combinations in the set of test data bit combinations to known correct check bits for each of the n n-bit data bit combinations, and indication means for generating an indication as to whether the check bits output by the error correction/detection logic are correct based on an output of the comparison means. As discussed above, one embodiment of such a tester 200 including test data bit generator 202, comparison logic

206 and verification signal 220 is shown in FIG. 4A and described in the specification at p. 14, line 25 – p. 15, line 7 and p.18, line 9 – p. 19, line 16.

Independent claim 27 is directed to a method of testing error correction/detection logic, the method comprising providing a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word; in response to said providing, the error correction/detection logic decoding the set of m+1 test code words; and verifying that the error correction/detection logic correctly decoded each of the m+1 test code words. One embodiment of such a method is shown in FIG. 7 and described in the specification at p. 22, line 29 – p. 24, line 8. In blocks 701-703, one member of a set of test code words is generated as a correct code word, and a single-bit error is introduced into the remaining code words such that single-bit errors are collectively represented at each bit position in the set of test code words. At 705, the generated set of test code words is provided for decoding to a decoder, such as decoder 300 of FIG. 6A or decoder 300A of FIG. 8. At 707, verification of whether the decoder correctly decoded each test code word occurs. For example, comparison logic 206 of tester 200 may determine whether decoder 300/300A corrected each test code word having a single-bit error and did not modify the test code word generated as a correct code word.

Independent claim 30 is directed to a method of testing error correction/detection logic, the method comprising providing a set of test code words to the error correction/detection logic, wherein said providing comprises introducing an error into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words; in response to said providing, the error correction/detection logic decoding each test code word in the set of test code words; and verifying that the error correction/detection logic correctly identified the error in each of the test code words. One embodiment of such a method is shown in FIG. 9 and described in the specification at p. 27, lines 4-15 in conjunction with FIGs. 8A-B and p. 24, line 10 – p. 27, line 2. At 901, check bit generator 102A and syndrome generator 304A are configured to introduce errors into test code words, producing each unused syndrome by substituting check bits corresponding to unused syndromes for correct check bits. The resulting test code words are then provided to error detection/correction unit 306. for decoding. At 905, verification of whether error detection/correction unit 306 correctly identified the introduced errors occurs.

Independent claim 34 is directed to a data processing system comprising a storage array comprising at least one mass storage device, a host computer system coupled to provide data to the storage array; and error correction/detection logic configured to generate check bits for the data being provided to the storage array. The host computer system is configured to test the error correction/detection logic by providing each of a set of n data bit combinations to the error detection/correction logic, wherein each data bit combination has n bits, wherein each possible value of each data bit is present in at least one of the n data bit combinations, wherein the set of n data bit combinations provided to the error detection/correction logic is a subset of a set of all data bit combinations that it is possible to create using n bits. In response to being provided with the set of n data bit combinations, the error detection/correction logic is configured to generate a set of check bits for each of the n data bit combinations. The host computer system is also configured to compare the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations. One embodiment of such a data processing system is illustrated in FIG. 2 and described in the specification at p. 9, line 22 – p. 11, line 12. Specifically, data processing system 2000 is shown to include a host 2001 coupled to provide data to a storage system 2003 including an array 2008 of mass storage devices 2010. As shown in FIGs. 4A and 4D and described in the specification at p. 15, lines 17-29, test data bit generator 202 may be configured to provide each of a set of n data bit combinations to check bit generator 102, where each possible value of each data bit is present in at least one of the n data bit combinations. Check bit generator 102 may then generate a set of check bits for each of the n data bit combinations, which may then be compared and verified by tester 200. As described above with respect to claim 13, the generation of the n data bit combinations and verification of check bits produced from those combinations that is described with respect to tester 200 may also be performed in software executable on host 2001.

Independent claim 35 is directed to a method of testing error detection/correction logic. The method includes the following: providing a subset of possible data bit combinations of n data bits to the error detection/correction logic, wherein the subset comprises n data bit combinations, wherein each possible value of each data bit is present in at least one of the n data bit combinations in the subset; verifying the error detection/correction logic by comparing a set of check bits generated by the error detection/correction logic for each of the n data bit combinations in the subset with a set of known correct check bits; and providing a first set of m+1 test code words to the error detection/correction logic, wherein a first test code word is a correct test code word and where each other test code word in the set of m+1 test code words comprises a single-bit error, wherein each test code word having a single-bit error has the single-bit error at a different bit position than each other test code word that has a single-bit error. Verifying the error detection/correction logic further comprises comparing a first output of the error detection/correction logic generated in response to said providing a first set of m+1 test code words with a first known correct output for each of the m+1 test code words. The method also includes providing a second set of test code words to the error detection/correction logic, wherein each test code word in the second set comprises an error introduced by substituting check bits corresponding to an unused syndrome for a correct set of check bits within a correct code word, wherein each test code word in the second set comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the second set of test code words; and verifying further comprises comparing a second output of the error detection/correction logic generated in response to said providing a second set of test code words with a second known correct output for each of the m+1 test code words. The method also includes indicating whether the error detection/correction logic is operating properly in response to said verifying. It is noted that claim 35 recites a combination of features that are separately recited in claims 27, 30 and 34. Support for claim 35 may be found in references made to the drawings and specification to support the aforementioned claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 3-5, 24, 25 and 27-29 are rejected under 35 U.S.C. § 102(b) as being anticipated by Kurihara (U.S. Patent No. 4,107,649) (hereinafter "Kurihara").

2. Claim 30 is rejected under 35 U.S.C. § 102(b) as being anticipated by Arroyo et al. (U.S. Patent No. 5,502,732) (hereinafter "Arroyo").

3. Claim 2 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Nielson et al. (U.S. Patent No. 5,619,642) (hereinafter "Nielson").

4. Claims 6-12, 26, 31-33 and 35 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Arroyo.

5. Claims 13-16 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Fielder et al. (U.S. Patent No. 6,446,037) (hereinafter "Fielder").

6. Claims 17-23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Fielder and further in view of Arroyo.

7. Claim 34 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Vishlitzky et al. (U.S. Patent No. 5,809,332) (hereinafter "Vishlitzky").

VII. ARGUMENT

First Ground of Rejection:

Claims 1, 3-5, 24, 25 and 27-29 are finally rejected under 35 U.S.C. § 102(b) as being anticipated by Kurihara. Appellant traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 1:

Kurihara fails to teach or suggest all of the limitations of Appellant's claim 1. Specifically, **Kurihara does not disclose a method of testing error correction/detection logic including creating an initial data bit combination having n bits, wherein each data bit in the initial data bit combination has a same logical value as each other data bit in the initial data bit combination, and shifting a first bit having a different logical value than the same logical value across the initial data bit combination, wherein each time the first bit is shifted, one of n data bit combinations is generated.** In contrast, Kurihara teaches a system in which a shift register sequentially shifts input data and feeds back output data and in which a parity signal is generated and compared to a predicted, or counted, parity signal to determine whether the shift register is working correctly. Specifically, Kurihara teaches that input data is shifted through a shift register while the number of logical values of '1' in the input data are counted and the number of logical values of '1' outputted by the shift registers is also counted (Kurihara, column 3, lines 13-24). Additionally, Kurihara's shift register performs a parity calculation on the data in the shift register. Kurihara then uses the number of 1s counted in the input data and/or the number of 1s counted in the output data to predict the parity value for the data in the shift register (Kurihara, column 3, lines 25-38). Kurihara's system includes a check circuit to compare the predicted or counted parity value with the parity value generated by the shift register and generates an error signal if they do not match (Kurihara, column 4, lines 13-19). However, Kurihara does not teach shifting a first bit having a different logical value across the initial data bit combination, wherein each time the bit is shifted, one of n data bit combinations is generated and provided to error detection/correction logic. While Kurihara does use a shift register and does shift input data through the shift register, Kurihara does not teach generating one of n data bit combinations each time the first bit is shifted and providing each of the bit combinations to error detection/correction logic. Kurihara only teaches that input data is shifted in to the shift register, without disclosing the additional limitations recited in Appellant's claim.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and asserts that "Kurihara teaches an error detection circuit which comprises a shift register... which sequentially shifts input data... [and that] the output from each stage of the shift register 101 is supplied to the parity generator 4, and a parity signal PARITY 1 is derived therefrom...." In response, Appellant asserts that Kurihara does not teach or suggest the specific acts that are required by Appellant's claim. Appellant's claim specifically recites creating a data bit combination such that each data bit of an initial data bit combination has a same logical value as each other bit, and shifting a bit having a different logical value than the same logical value across the bit combination. Whether Kurihara's structure could or could not hypothetically be configured to perform these acts in hindsight is irrelevant to the question of anticipation of Appellant's method claim. As noted below, anticipation requires that Kurihara disclose the acts recited in Appellant's claim, arranged as in the claim, in as complete detail as recited in the claim. Kurihara simply fails to do so.

In further regard to claim 1, **Kurihara fails to teach providing each of the n data bit combinations to the error detection/correction logic; in response to said providing, the error detection/correction logic generating a set of check bits for each of the n data bit combinations; comparing the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations.** Instead, Kurihara teaches only that input data enters the input end of the shift register (Kurihara, column 3, lines 13-15). Kurihara mentions nothing about providing each of n data bit combinations to the error detection/correction logic. Kurihara does not mention anything about the size of input data nor about the input data including n data bit combinations. Kurihara is equally silent regarding the error detection/correction logic generating a set of check bits. Instead, Kurihara's shift register calculates the single parity bit from the input data in the shift register. Calculating a single parity bit is very different from generating a *set of check bits.* **Kurihara also does not disclose comparing the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each for the n data bit combinations.** In contrast, Kurihara compares a generated parity bit with a predicted, or counted, parity bit from the input data (and/or output data). The counted parity value of Kurihara is clearly not a *known correct set of check bits.* Furthermore, Kurihara does not teach comparing the set of check bits with a known correct set of check bits *for each of the n data bit combinations.* Kurihara only teaches that input data is shifted into the shift register and that the number of logical values of 1 in the input data are counted. Kurihara does not mention n data bit combinations nor does he disclose comparing check bits with known correct check bits *for each of the n data bit combinations.*

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and asserts that Kurihara teaches comparing a predicted parity value with the parity value produced by the error detection circuit. However, as argued above, generation of <u>a single parity bit</u> is not in any way equivalent to <u>generating a set of check bits</u>, as recited in Appellant's claim. Further, Kurihara does not teach a comparison against a <u>known correct set of check bits</u>. **Kurihara's predicted parity bit is not a known correct parity bit.** Rather, it is simply a parity bit generated through redundant means independent of the main error detection circuit. The fact that a parity bit is generated redundantly does not entail that the resulting parity bit is known to be correct.

More generally, **<u>Kurihara does not specifically condition input data to an error detection circuit in any way for the purpose of testing that circuit.</u>** Rather, Kurihara passively checks the result computed for every input, regardless of how the input was created, against a redundantly-generated result. The primary observation of

Kurihara consists in noting that for a specific type of linear-feedback shift register, the redundant result may be generated through one of two simple exclusive-OR (XOR) functions of the input bits, depending on whether the number of feedback loops within the shift register is even or odd (Kurihara, col. 2, lines 22-67). However, this has nothing whatsoever to do with the testing method recited in Appellant's claims, in which properties of test inputs are specifically circumscribed. In fact, Kurihara's aim in leveraging this specific property of this particular shift register embodiment is to simplify the generation of his redundant, brute-force testing apparatus, precisely so that he can ignore the problem of how to condition the test inputs to the apparatus as addressed in Appellant's claims. That is, by taking advantage of a property of a particular type of error detection circuit to simplifying a brute-force implementation for testing that circuit in which any input may be redundantly computed for correctness, Kurihara has no need to consider the economics of testing for other circuits that do not yield to the same brute-force approach. Kurihara is directed to a completely different type of solution than that recited in Appellant's claims.

Appellant notes that anticipation requires the presence in a single prior art reference disclosure of each and every limitation of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). For at least the foregoing reasons, Kurihara clearly fails to meet this standard, and Appellant submits that Kurihara consequently fails to anticipate claim 1.

Claim 3:

In addition to the reasons given above for claim 1, Kurihara fails to teach or suggest all of the limitations of claim 3. Specifically, as argued in greater detail below with respect to claim 27, Kurihara fails to teach or suggest a method of testing error correction/detection logic that includes providing a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word, and where the method further includes the error correction/detection logic decoding the set of m+1 test code words in response to said providing, and verifying that the error correction/detection logic correctly decoded each of the m+1 test code words.

Claims 4 and 5:

In addition to the reasons given above for claim 3, Kurihara fails to teach or suggest the limitations recited in claim 4. Specifically, Kurihara fails to disclose the method as claimed in which each bit in the first test code word has a same logical value and each test code word other than the first test code word comprises a bit having an opposite logical value at the different bit position at which a single-bit error is located.

As argued above with respect to claim 1, Kurihara simply does not disclose the detailed constraints regarding the generation of input data or test code words to be applied to error correction/detection logic for the purpose of testing that logic. That is, Kurihara does not disclose deliberately constraining the relationships among bits within a set of test code words such that bits within a first test code word have the same logical value and bits of other test code words, each including a single-bit error, have an opposite logical value at the location of the single-bit error.

Claim 24:

Regarding claim 24, Kurihara fails to teach test check bit generating means for creating a set of test data bit combinations and providing the set of test data bit combinations to a check bit generator comprised in the error correction/detection logic. In contrast, as discussed above, Kurihara teaches inputting data to a shift register. Kurihara does not disclose creating a set of test data bit combinations and providing the set of test data bit combinations to a check bit generator. The Examiner cites FIG. 2 and column 5, lines 7-34 of Kurihara. The cited portions of Kurihara only describe shifting input data into a shift register, generating a parity value for the input data in the shift register and counting the values of 1 in the input data for comparison. The Examiner appears to be implying that Kurihara's input data is a set of test data bit combinations and that Kurihara's shift register is a check bit generator. However, Kurihara does not teach that the input data to the shift register is a set of test data bit combinations created by test check bit generating means.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and asserts that Kurihara teaches an error detection circuit including a shift register for receiving input data and an output stage for feeding back output data, as well as a parity generator for generating a parity signal from the data stored in the shift register. However, as argued above, Kurihara teaches only that a parity bit may be generated in several redundant ways (parity generator 4, counters 2 and 3) for a given set of data bits resident within a shift register. Kurihara does not suggest in any way that the data bits resident within the shift register are test data bit combinations created by test check bit generating means. In fact, Kurihara is entirely silent as to the source of the input data stored within the shift register.

Further regarding claim 24, Kurihara also fails to disclose that the set of test data bit combinations comprises n n-bit data bit combinations, wherein each possible value of each data bit is present in at least one of the n n-bit data bit combinations in the set of test data bit combinations. Nowhere does Kurihara mention anything about a set of test data bit combinations comprising n n-bit data bit combinations and the Examiner's cited passages (FIG 2 and column 5, lines 7-34) only describe shifting input data into a shift register, generating a parity value for the data in the shift register and counting the values of 1 in the input data for comparison. Neither the cited portions, nor the remainder of Kurihara, mention anything regarding n n-bit data combinations. Furthermore, nowhere does Kurihara teach that each possible value of each data bit is present in at least one of the n n-bit data bit combinations of test data bit combinations. Kurihara only refers generically to "input data" without describing anything about any combinations of bits in the makeup of the input data (see, Kurihara, column 1, lines 44-48, and column 3, lines 13-17). As argued above with respect to claim 1, Kurihara is simply not directed to the problem of generating test bits, patterns or other data that function as *inputs* to error correction/detection logic. Rather, Kurihara discloses a brute-force implementation for redundantly computing an error detection value for **any input** provided to an error detection circuit, avoiding any motivation for constraining the test inputs in any way.

Additionally, Kurihara fails to teach comparison means for comparing check bits output by the error/detection logic for each of the n n-bit data bit combinations in the set of test data bit combinations to known correct check bits for each of the n n-bit data bit combinations. Kurihara only teaches comparing a one-bit predicted parity value, generated by counting the "1"s in input data, with a one-bit parity value generated on the data in the shift register. Nowhere does Kurihara disclose anything regarding comparing check bits for each of n n-bit data bit combinations. As noted above, Kurihara fails to mention n n-bit data bit combinations at all.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and asserts that Kurihara teaches a check circuit configured to compare the predicted parity value with the actual parity value. However, as argued above with respect to claim 1, Kurihara does not teach the generation of check bits corresponding to each of several data bit combinations and comparing the generated check bits against known correct check bits. Further, as argued above, Kurihara fails to teach or suggest that the n-bit data bit combinations are included as part of a set of test bit data combinations.

Kurihara clearly fails to meet the standard of anticipation described above with respect to claim 24. Therefore, Appellant submits that Kurihara does not anticipate claim 24.

Claim 27:

Regarding claim 27, Kurihara does not teach a method including providing a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word. The Examiner's cited portions (FIG 2, column 2, lines 61-68 and column 3, lines 13-34) only describe how Kurihara calculates a predicted parity value and compares it to a generated parity value to detect malfunction of the error detection circuit. However, nowhere in the cited passages does Kurihara mention providing a set of m+1 test code words to the error correction/detection logic. The Examiner is apparently equates Kurihara's references to "input data" shifted into a shift register as teaching providing a set of m+1 test code words to the error correction/detection logic. However, Kurihara never mentions a set of m+1 test code words, nor does he describe his "input data" as test code words. Further, Kurihara does not describe his input data as having m bits. In fact, Kurihara never mentions anything about the size of input data. Thus, Kurihara's input data cannot be equated to a set of m+1 test code words, wherein each code word has m bits, as the Examiner contends. As argued above with respect to claims 1 and 24, Kurihara does not disclose any aspect of conditioning or constraining test code word inputs to error correction/detection logic in any way. Kurihara is instead directed to the *opposite* approach of not constraining test code word inputs, but rather redundantly checking every output produced by his error detection logic.

Additionally, Kurihara fails to disclose wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word. Kurihara's method simply does not include a set of m+1 test code words that includes a first test code word that is a correct code words and wherein the other test code words each include a single-bit error at different bit positions. Kurihara teaches a completely different type of method incompatible with a set of test code words. Kurihara's entire method is only concerned with comparing predicted and generated parity values. It would not even make sense to use a set of m+1 test code words where every code word other than a first correct code word includes single bit errors in Kurihara's method, because his method would merely compare the predicted parity value (total counted "1"s) with a generated parity value (by a parity generator). Such a method has no way to detect or make use of single-bit errors in test code words.

Furthermore, Kurihara's method clearly does not include the error correction/detection logic <u>decoding the set of m+1 test code words</u>. As noted above, Kurihara does not teach providing a set of m+1 test code words to the error correction/detection logic. Kurihara also fails to teach error correction/detection logic that decodes such a set of m+1 test code words. Kurihara's entire system includes only a shift register, 2 one-bit counters, a parity generator, and a check circuit (FIGs 1, 2, and column 3, lines 39-65). There is nothing in Kurihara's system that is described as capable of decoding a set of m+1 test code words.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and asserts that Kurihara teaches an error detection circuit including a shift register that sequentially shifts input data and feeds back output data to desired shift register stages, as well as a parity generator that generates a parity signal in accordance with data stored in the shift register. The Examiner further asserts that "the input data and the shift register provides the test code words to the error correction/detection logic." Appellant disagrees with the Examiner's interpretation of Kurihara. As argued above, Appellant's claim 27 specifically recites test code words having definite characteristics, such as the positioning of single-bit errors within the test code word. Kurihara does not disclose <u>any aspect</u> of specifically positioning single-bit errors within a set of test code words. In fact, as argued above with respect to claim 24, Kurihara does not teach that the input to the shift register is configured in any deliberate way for testing of error detection logic or for other purposes. Rather, Kurihara merely computes parity in two different ways and assumes that if the two sources of parity disagree, an error exists. Kurihara makes no teaching whatsoever regarding deliberately introducing errors into any portion of the shift register or parity generation logic, as recited in claim 27.

For at least the foregoing reasons, Appellant submits that Kurihara fails to anticipate claim 27.

Claims 28 and 29:

In addition to the reasons given above for claim 27, Appellant submits that Kurihara does not anticipate claim 28 for at least the reasons given above with respect to claim 4.

Second Ground of Rejection:

Claim 30 is rejected under 35 U.S.C. § 102(b) as being anticipated by Arroyo. Appellant traverses this rejection for the following reasons.

Arroyo fails to teach or suggest all of the limitations of claim 30. Specifically, Arroyo does not teach a method including providing a set of test code words to the error correction/detection logic, wherein said providing comprises introducing an error into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words. Arroyo teaches a system for checking the test logic contained in a computer memory system during POST (Abstract). The Examiner's cited passage (Arroyo, column 5, line 55-column 6, line 4) describes how a CPU writes data including two bits set to "1" to memory and how a diagnostic bit controlling multiplexer 33 replaces the two bits set to "1" with zeros. In Arroyo's system ECC generator 31 generates check bits for the unmodified data and stores those check bits in memory. When the data is read from memory (including the zeros substituted by multiplexer 33), ECC generator 41 generates check bits that does not match the check bits originally generated by ECC generator 31 when the data was stored. Hence, Arroyo does not substitute check bits corresponding to an unused syndrome, but rather teaches replacing two bits set to "1" with bits set to "0". Arroyo's technique is not equivalent to that recited in claim 30.

Further, Arroyo's method does not include providing a set of test code words, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome. In contrast, Arroyo teaches that "a multiplexer is provided in the data write path which substitutes *a constant set of identical bits* for the actual data generated by the CPU" (emphasis added, Arroyo, column 2, lines 22-25). Arroyo clearly teaches substituting a constant set of identical bits and thus clearly teaches away from providing a set of test code words wherein each test code word comprises substituted check bits corresponding to a different unused syndrome. That is, Arroyo's substituting a constant set of identical bits for actual data, without regard to a further constraint, not only does not suggest but is incompatible with deliberately substituting check bits for a given test code word that are specified to correspond to different unused syndromes from those specified for other test code words.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and cites Arroyo's description of testing ECC logic at col. 7, lines 22-44. However, Appellant notes that as argued above, Arroyo does not teach introducing an error into a code word by substituting check bits corresponding to an unused syndrome for a correct set of check bits in the code word, as recited in claim 30. Rather, Arroyo introduces an error into a code word simply by zeroing out the check bits read from memory. The resulting syndrome of Arroyo is thus entirely dependent upon the check bits computed from the data read from memory, since the syndrome is formed from the logical XOR of the computed check bits and the check bits read from memory, but a logical XOR of any value with the zero is simply the value itself. Since Arroyo is substituting only zeroes for the check bits read from memory, Arroyo simply cannot function to deliberately substitute check bits corresponding to an unused syndrome, as required by Appellant's claim 30. Instead, whether a resulting syndrome of Arroyo is unused depends upon the data read from memory. But Arroyo does not teach any substitution of even the data bits in this case: the data written to memory at step 4 of

FIG. 5a is the same as the data read from memory at step 5 of FIG. 5b, which is then used to generate the check bits compared with the check bits that have been forced to zero. Thus, Arroyo does not teach any aspect of substituting either check bits or data bits corresponding to an unused syndrome. As argued previously, the technique of Arroyo is fundamentally dissimilar from the technique recited in claim 30.

For at least the reasons given above, Appellant submits that Arroyo fails to anticipate claim 30.

Third Ground of Rejection:

Claim 2 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Nielson. Appellant traverses this rejection for at least the reasons given above with respect to claim 1. Nielson contains no teachings that overcome any of the deficiencies of Kurihara noted above in regard to claim 1.

Fourth Ground of Rejection:

Claims 6-12, 26, 31-33 and 35 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Arroyo. Appellant traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 6 and 7:

Appellant submits that claims 6 and 7 are distinguishable for at least the reasons given above with respect to claims 1 and 3. Arroyo contains no teachings that overcome any of the deficiencies of Kurihara noted above in regard to claims 1 and 3.

Claim 8:

In addition to those reasons given above with respect to claims 1 and 3, Kurihara in view of Arroyo fails to teach or suggest the limitation of claim 8 in which verifying that the error correction/detection logic correctly decoded each of the m+1 test code words, as recited in claim 3, comprises verifying that the error correction/detection logic indicates that the first test code word is correct in response to being provided with the first test code word. In rejecting claim 8 in the Final Action, the Examiner relies on Arroyo, col. 8, lines 19-41 to teach this limitation. However, the cited portion of Arroyo corresponds to claim 1 of Arroyo, which is directed solely to inducing errors within bit patterns and determining whether an ECC system detects the induced errors. This is in no way suggestive of <u>deliberately generating a first test code word as a correct code word</u>, as recited in Appellant's claim 3, and <u>verifying that the first test code word is indicated as being correct</u>, as recited in claim 8. In other words, the cited portion of Arroyo is directed to the correct detection of errors, not the correct detection of non-errors. The two cases are <u>not equivalent</u>, as it is possible for error correction/detection logic to falsely indicate that data is incorrect when it is in fact correct, <u>in addition to</u> the case of falsely indicating that data is correct when it is in fact incorrect. Testing for the latter condition does not ensure that the former will not occur. Thus, Appellant submits that claim 8 is distinguishable over the cited art.

<u>Claims 9 and 26:</u>

Appellant notes that claims 9 and 26 recite limitations similar to those recited in claim 30. Appellant submits that claims 9 and 26 are distinguishable for at least the reasons given above with respect to claims 1, 25 and 30.

<u>Claims 10-12:</u>

In rejecting claim 10, the Examiner asserts that Kurihara teaches creating an initial code word wherein each bit in the initial code word has a same logical value. However, as argued above with respect to claim 1, Kurihara does not teach or suggest any aspect of creating test code words that are deliberately specified or constrained in any way for the purpose of testing error correction/detection logic. Therefore, Kurihara cannot teach the limitation recited in claim 10. For at least these reasons in addition to those given above for claims 1 and 9, Appellant submits that claims 10-12 are distinguishable over the cited art.

Claims 31-33:

In rejecting claim 31, the Examiner asserts that Kurihara teaches creating an initial code word wherein each bit in the initial code word has a same logical value. However, as argued above with respect to claims 1 and 10, Kurihara does not teach or suggest any aspect of creating test code words that are deliberately specified or constrained in any way for the purpose of testing error correction/detection logic. Therefore, Kurihara cannot teach the limitation recited in claim 31. For at least these reasons in addition to those given above for claim 30, Appellant submits that claims 31-33 are distinguishable over the cited art.

Claim 35:

Regarding claim 35, Kurihara in view of Arroyo fails to teach providing a subset of possible data bit combinations of n data bits to the error detection/correction logic, wherein the subset comprises n data bit combinations, <u>wherein each possible value of each data bit is present in at least one of the n data bit combinations in the subset</u>. As noted above regarding claim 1, Kurihara teaches comparing a one bit predicted parity value with a one bit generated parity value. Arroyo teaches substituting "a constant set of identical bits [zeros]" (Arroyo, column 2, lines 22-25). Neither Kurihara nor Arroyo teach providing a subset of data bit combinations of n data bits, wherein each possible value of each data bit is present in at least one of the n data bit combinations. Kurihara only teaches input data without mentioning anything regarding possible data bit combinations of n data bits and Arroyo teaches the use of a constant set of identical bits.

Additionally, Kurihara in view of Arroyo also fails to teach verifying the error detection/correction logic <u>by comparing a set of check bits generated</u> by the error detection/correction logic <u>for each of the n data bit combinations</u> in the subset <u>with a set of known correct check bits</u>. The Examiner contends that Kurihara teaches this.

However, the Examiner's cited portions of Kurihara (FIG 2, column 1, lines 44-60 and column 2, lines 12-19) only refer to comparing a predicted parity value with a generated parity value for input data in a shift register. Nowhere does Kurihara mention anything about comparing a set of check bits generated by the error detection/correction logic for each of n data bit combinations. In fact, as noted above, Kurihara fails to mention n data bit combinations as all and certainly does not disclose comparing a set of check bits generated for each of n data bit combinations with a set of known correct check bits. Arroyo also fails to teach such functionality. In contrast, as noted above, Arroyo teaches substituting a constant set of identical bits (zeros) when writing data to memory and comparing check bits generated on both the original data and the data with zeros substituted. Since neither Kurihara nor Arroyo teaches comparing check bits generated for each of n data bit combinations with a set of known correct check bits, no combination of Kurihara and Arroyo would include such a feature.

Kurihara in view of Arroyo also fails to teach <u>providing a first set of m+1 test code words to the error detection/correction logic</u>, wherein a <u>first test code word is a correct test code word</u> and where <u>each other test code word</u> in the set of m+1 test code words <u>comprises a single-bit error</u>, wherein each test code word having a single-bit error has the single-bit error <u>at a different bit position than each other test code word</u> that has a single-bit error. The Examiner asserts that Kurihara teaches such functionality in his method and cites Fig 2, column 2, lines 61-68 and column 3 lines 13-34. However, as noted above regarding the rejection of claim 27, these cited portions fail to mention anything about a set of m+1 test code words and the remarks and arguments presented above regarding claim 27 also apply here. Arroyo also fails to teach providing a set of m+1 test code words wherein a first test code word is a correct test code word and where each other test code word comprises a single bit error at a different bit offset than each other test code word. As noted above, Arroyo teaches that "a multiplexer is provided in the data write path which substitutes *a constant set of identical bits* for the actual data generated by the CPU" (emphasis added, Arroyo, column 2, lines 22-25). Arroyo further teaches an ECC generator 31 that generates check bits for the unmodified data and stores those check bits in memory. When the data is read from memory (including the zeros substituted by multiplexer 33), ECC generator 41 generates check bits that does not match the check bits originally generated by ECC generator 31 when the data was stored. Hence, Arroyo does not provide a set of m+1 test code words, but rather teaches comparing check bits generated for a set of data with check bits generated on the same data after having zeros inserted for certain bits. Thus, Arroyo clearly fails to teach providing a set of m+1 test code words wherein a first test code word is a correct test code word and where each other test code word comprises a single bit error at a different bit offset than each other test code word. Additionally, no combination of Kurihara and Arroyo can include such a feature.

Kurihara in view of Arroyo also fails to teach providing <u>a second set of test code words</u> to the error detection/correction logic, wherein each test code word in the second set comprises an error introduced by <u>substituting check bits corresponding to an unused syndrome</u> for a correct set of check bits within a correct code word, wherein each test code word in the second set comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the second set of test code words. Kurihara clearly fails to teach <u>substituting check bits corresponding to an unused syndrome</u> and the Examiner relies upon Arroyo for this feature. However, Arroyo also fails to teach substituting check bits corresponding to an unused syndrome. Please see above regarding claim 30 for remarks regarding how Arroyo fails to teach substituting check bits corresponding to an unused syndrome.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and reiterates numerous aspects of Kurihara and Arroyo discussed above with respect to other claims. Appellant makes specific reference to the arguments set forth above with respect to claims 27 and 30, each of which include limitations also present in claim 35. As argued above, Kurihara and Arroyo fail to anticipate these respective claims, and neither reference suggests the limitations that the other fails to teach. Therefore, the combination of Kurihara and Arroyo cannot teach or suggest the combination recited in claim 35. Consequently, Appellant submits that claim 35 is distinguishable.

Fifth Ground of Rejection:

Claims 13-16 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Fielder. Appellant traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claim 13:

Kurihara in view of Fielder fails to teach all of the limitations recited in claim 13, including a computer readable medium comprising program instructions computer executable to: create an initial data bit combination having n bits, wherein each data bit in the initial data bit combination has a same logical value as each other data bit in the initial data bit combination; shift a first bit having an different logical value than the same logical value across the initial data bit combination, wherein each time the first bit is shifted, one of n data bit combinations is generated; provide each of the n data bit combinations to error detection/correction logic; compare a set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations; and dependent on an outcome of said comparing, generate an indication of whether the error detection/correction logic correctly generated the set of check bits. Appellant notes that the remarks above regarding claim 1 in view of Kurihara also apply to claim 13.

Fielder teaches a method for scalable coding of audio data into a core layer in response to a desired noise spectrum established according to psychoacoustic principles that supports coding augmentation data into augmentation layers in response to various criteria (Fielder, Abstract). Fielder's method has nothing to do with the error detection/correction circuit of Kurihara. The Examiner is relying upon Fielder to teach a computer readable medium. However, the respective inventions of Kurihara and Fielder are directed to completely different fields of endeavor. Fielder's scalable audio encoding process has absolutely no relevance to Kurihara's error detection circuit checking circuit. Kurihara teaches a hardware circuit for detecting malfunction of an error detection circuit and does not teach anything related to Fielder's scalable audio encoding software. The Examiner's cited passage from Fielder (column 4, lines 60-64) describes how Fielder's scalable audio encoding and decoding processes may be conveyed by a machine readable medium. However, Kurihara's teachings are <u>specific to a hardware circuit implementation</u>. There is no suggestion to in either Fielder or Kurihara to create a program instruction computer readable medium implementation of Kurihara's circuit. In fact, since Kurihara's teachings are specific to a hardware circuit, it is unclear how Kurihara's teachings could even be applied to a program instruction computer readable medium implementation.

Moreover, the Fielder reference is not analogous art. "In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." *In re Oeticker*, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). "A reference is reasonably pertinent if, even though it may be in a different field from that of the inventor's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to an inventor's attention in considering his problem." *In re Clay*, 966 F.2d 656, 659, 23 USPQ2d 1058, 1060-61 (Fed. Cir. 1992). Here, Fielder is clearly not in the field of Appellant's endeavor of testing error correction/detection logic. In contrast, Fielder deals with scalable encoding and decoding of audio data (Fielder, col. 2, lines 35-49). Furthermore, the subject of Fielder would not logically have commended itself to an inventor's attention when considering the problem addressed by Appellant. One of skill in the art seeking to address the problem of testing error correction/detection logic would not have any logical reasons for considering a technique used to encode and decode audio data. Thus, Fielder is not within Appellant's field of endeavor and is not pertinent to the problem addressed by Appellant's invention. Nor is Fielder within the field of endeavor or pertinent to the problem addressed by Kurihara. Accordingly, Fielder is non-analogous art and cannot properly be combined with Kurihara.

The Examiner states that Fielder is in an analogous art. However, the Examiner has clearly over-generalized the meaning of "analogous art." *In re Oeticker* refers to Appellant's field of endeavor and particular problem. The analogous art requirement can always be made meaningless by over-generalizing the field of endeavor or problem. Almost any art may be considered pertinent if the problem is stated in general enough terms. That is why the courts have insisted that art used in § 103 rejections be in the "same field of endeavor" or "pertinent to the particular problem." Fielder pertains to scalable encoding and decoding of audio data, which has nothing to do with Appellant's field of endeavor or particular problem.

In the "Response to Arguments" section of the Final Action, the Examiner disagrees with the foregoing and reasserts that Fielder teaches a computer readable medium. However, the Examiner's assertions do nothing to address Appellant's contention that Fielder is not analogous art with respect to Appellant's claimed invention. As argued above, the problem addressed by Fielder, that of audio encoding, has nothing whatsoever to do with the problem of testing an error detection/correction system. Appellant maintains that one of skill in the art in testing error detection systems would have no motivation whatsoever to consult art relating to audio encoding methods. Moreover, Appellant refers to the additional arguments made above with respect to claim 1 in support of the contention that Kurihara fails to teach or suggest similar limitations of claim 13.

Therefore, for at least the reasons given above, the rejection of claim 13 is not supported by the prior art, and Appellant submits that claim 13 is distinguishable.

Claim 14:

Appellant notes that claim 14 recites limitations similar to those recited in claims 3 and 30, and notes that Fielder does not teach or suggest those limitations of claim 14 that Kurihara fails to disclose. For at least the reasons given above with respect to claims 13, 3 and 30, Appellant submits that claim 14 is distinguishable.

<u>Claims 15 and 16:</u>

Appellant notes that claims 15 and 16 recite limitations similar to those recited in claims 4 and 5, and notes that Fielder does not teach or suggest those limitations of claims 15 and 16 that Kurihara fails to disclose. For at least the reasons given above with respect to claims 14 and 4, Appellant submits that claims 15 and 16 are distinguishable.

<u>Sixth Ground of Rejection:</u>

Claims 17-23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Fielder and further in view of Arroyo. Appellant traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

<u>Claims 17 and 18:</u>

Appellant notes that claims 17 and 18 recite limitations similar to those recited in claims 6 and 7. For at least the reasons given above with respect to claim 13 and claims 6 and 7, Appellant submits that claims 17 and 18 are distinguishable.

<u>Claim 19:</u>

Appellant notes that claim 19 recites limitations similar to those recited in claim 8. For at least the reasons given above with respect to claims 13 and 8, Appellant submits that claim 19 is distinguishable.

<u>Claim 20:</u>

Appellant notes that claim 20 recites limitations similar to those recited in claims 9 and 30. For at least the reasons given above with respect to claim 13 as well as claims 9 and 30, Appellant submits that claim 20 is distinguishable.

Claims 21-23:

Appellant notes that claims 21-23 respectively recite limitations similar to those recited in claims 10-12. For at least the reasons given above with respect to claims 20 and claims 10-2, Appellant submits that claims 21-23 are distinguishable.

Seventh Ground of Rejection:

Claim 34 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Kurihara in view of Vishlitzky. Appellant traverses this rejection for the following reasons.

Kurihara in view of Vishlitzky fails to teach providing each of a set of n data bit combinations to the error detection/correction logic. Kurihara mentions nothing about providing each of n data bit combinations to the error detection/correction logic. Kurihara does not mention anything about the size of input data nor does he mention input data including a set of n data bit combinations. The Examiner has not cited any passage of Kurihara that discusses a set of n data bit combinations. As argued above with respect to claim 1, Appellant notes that Kurihara fails to teach or suggest any aspect of providing specific combinations of test inputs to error correct/detection logic for testing. Also as argued with respect to claim 1, Appellant notes that Kurihara also fails to teach generating a set of check bits, as distinct from the single parity bit of Kurihara, corresponding to an input data bit combination. Kurihara further fails to teach comparing the generated check bits with a known correct set of check bits for each input data bit combination. Further, Vishlitzky, which the Examiner relies on to teach the storage array limitations of claim 34, fails to teach or suggest those limitations of claim 34 that are omitted by Kurihara.

For at least the reasons given above, Appellant submits that claim 34 is distinguishable over the cited references.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-35 was erroneous, and reversal of the Examiner's decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-94400/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

[illegible]

Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850
Date: August 19, 2005

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method of testing error correction/detection logic, the method comprising:

creating an initial data bit combination having n bits, wherein each data bit in the initial data bit combination has a same logical value as each other data bit in the initial data bit combination;

shifting a first bit having a different logical value than the same logical value across the initial data bit combination, wherein each time the first bit is shifted, one of n data bit combinations is generated;

providing each of the n data bit combinations to the error detection/correction logic;

in response to said providing, the error detection/correction logic generating a set of check bits for each of the n data bit combinations;

comparing the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations; and

dependent on an outcome of said comparing, generating an indication of whether the error detection/correction logic correctly generated the set of check bits.

2. The method of claim 1, wherein the error detection/correction logic comprises a Hamming code encoder and a Hamming code decoder.

3. The method of claim 1, further comprising:

providing a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word;

in response to said providing, the error correction/detection logic decoding the set of m+1 test code words; and

verifying that the error correction/detection logic correctly decoded each of the m+1 test code words.

4. The method of claim 3, wherein each bit in the first test code word has a same logical value, and wherein each test code word other than the first test code word comprises a bit having an opposite logical value at the different bit position.

5. The method of claim 4, wherein said providing a set of m+1 test code words comprises shifting the bit having the opposite logical value across the first test code word, wherein each time the bit is shifted, one of the test code words other than the first test code word is created.

6. The method of claim 3, wherein said verifying that the error correction/detection logic correctly decoded each of the m+1 test code words comprises verifying that the error correction/detection logic detects each of the single-bit errors.

7. The method of claim 3, wherein said verifying that the error correction/detection logic correctly decoded each of the m+1 test code words comprises

verifying that the error correction/detection logic corrects each of the single-bit errors that occurs in a data bit within each of the m+1 test code words.

8. The method of claim 3, wherein said verifying that the error correction/detection logic correctly decoded each of the m+1 test code words comprises verifying that the error correction/detection logic indicates that the first test code word is correct in response to being provided with the first test code word.

9. The method of claim 1, further comprising:

providing a set of test code words to the error correction/detection logic, wherein an error has been introduced into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words;

in response to said providing, the error correction/detection logic decoding each test code word in the set of test code words; and

verifying that the error correction/detection logic correctly identified the error in each of the test code words.

10. The method of claim 9, wherein said providing a set of test code words comprises creating an initial code word, wherein each bit in the initial code word has a same logical value.

11. The method of claim 10, wherein the same logical value is a logical 0, and wherein the substituted check bits in each code word equal a set of bits in one of the unused syndromes.

12. The method of claim 10, wherein the same logical value is a logical 1, and wherein each bit in the substituted check bits in each code word equals an inverse logical value of each bit in one of the unused syndromes.

13. A computer readable medium comprising program instructions computer-executable to:

create an initial data bit combination having n bits, wherein each data bit in the initial data bit combination has a same logical value as each other data bit in the initial data bit combination;

shift a first bit having a different logical value than the same logical value across the initial data bit combination, wherein each time the first bit is shifted, one of n data bit combinations is generated;

provide each of the n data bit combinations to error detection/correction logic;

compare a set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations; and

dependent on an outcome of said comparing, generate an indication of whether the error detection/correction logic correctly generated the set of check bits.

14. The computer readable medium of claim 13, further comprising program instructions computer-executable to:

provide a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word; and

verify that the error correction/detection logic correctly decoded each of the m+1 test code words.

15. The computer readable medium of claim 14, wherein each bit in the first test code word has a same logical value, and wherein each test code word other than the first test code word comprises a bit having an opposite logical value at the different bit position.

16. The computer readable medium of claim 15, further comprising program instructions computer-executable to shift the bit having the opposite logical value across the first test code word, wherein each time the bit is shifted, one of the test code words other than the first test code word is created.

17. The computer readable medium of claim 14, further comprising program instructions computer-executable to verify that the error correction/detection logic detects each of the single-bit errors.

18. The computer readable medium of claim 15, further comprising program instructions computer-executable to verify that the error correction/detection logic corrects each of the single-bit errors that occurs in a data bit within each of the m+1 test code words.

19. The computer readable medium of claim 14, further comprising program instructions computer-executable to verify that the error correction/detection logic indicates that the first test code word is correct in response to being provided with the first test code word.

20. The computer readable medium of claim 13, further comprising program instructions computer-executable to:

provide a set of test code words to the error correction/detection logic, wherein an error has been introduced into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words; and

verify that the error correction/detection logic correctly identified the error in each of the test code words.

21. The computer readable medium of claim 20, further comprising program instructions computer-executable to create an initial code word, wherein each bit in the initial code word has a same logical value.

22. The computer readable medium of claim 21, wherein the same logical value is a logical 0, and wherein the substituted check bits in each code word equal a set of bits in one of the unused syndromes.

23. The computer readable medium of claim 21, wherein the same logical value is a logical 1, and wherein each bit in the substituted check bits in each code word equals an inverse logical value of each bit in one of the unused syndromes.

24. A tester for testing error correction/detection logic, the tester comprising:

test check bit generating means for creating a set of test data bit combinations and providing the set of test data bit combinations to a check bit generator comprised in the error correction/detection logic, wherein the set of test data bit combinations comprises n n-bit data bit combinations, wherein each possible logical value of each data bit is present in at least one of the n n-bit data bit combinations in the set of test data bit combinations;

comparison means for comparing check bits output by the error correction/detection logic for each of the n n-bit data bit combinations in the set of test data bit combinations to known correct check bits for each of the n n-bit data bit combinations; and

indication means for generating an indication as to whether the check bits output by the error correction/detection logic are correct based on an output of the comparison means.

25. The tester of claim 24, further comprising:

test code word generating means for creating a set of test code words and providing the set of test code words to the error correction/detection logic, wherein the set of test code words comprises m+1 m-bit test code words, wherein a first test code word is a correct code word, and wherein each other test code word comprises a single-bit error, and wherein each of the other test code words comprises the single-bit error at a different bit position than any other of the other test code words;

wherein the comparison means are further for comparing an output of the error correction/detection logic with a known correct output for each test code word in the set of test code words.

26. The tester of claim 24, further comprising:

test code word generating means for creating a set of test code words and providing the set of test code words to the error correction/detection logic, wherein an error has been introduced into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words;

wherein the comparison means are further for comparing an output of the error correction/detection logic to a known correct output for each of the test code words.

27. A method of testing error correction/detection logic, the method comprising:

providing a set of m+1 test code words to the error correction/detection logic, wherein each code word has m bits, wherein a first test code word in the set of m+1 test code words is a correct code word, wherein each test code word other than the first test code word comprises a single-bit error at a different bit position within the code word than each other test code word;

in response to said providing, the error correction/detection logic decoding the set of m+1 test code words; and

verifying that the error correction/detection logic correctly decoded each of the m+1 test code words.

28. The method of claim 27, wherein each bit in the first test code word has a same logical value, and wherein each test code word other than the first test code word comprises a bit having an opposite logical value at the different bit position.

29. The method of claim 28, wherein said providing comprises shifting the bit having the opposite logical value across the first test code word, wherein each time the bit is shifted, one of the test code words other than the first test code word is created.

30. A method of testing error correction/detection logic, the method comprising:

providing a set of test code words to the error correction/detection logic, wherein said providing comprises introducing an error into each of the test code words in the set by substituting check bits corresponding to an unused syndrome for a correct set of check bits within each test code word, wherein each test code word comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the set of test code words;

in response to said providing, the error correction/detection logic decoding each test code word in the set of test code words; and

verifying that the error correction/detection logic correctly identified the error in each of the test code words.

31. The method of claim 30, wherein said providing comprises creating an initial code word, wherein each bit in the initial code word has a same logical value.

32. The method of claim 31, wherein the same logical value is a logical 0, and wherein the substituted check bits in each code word equal a set of bits in one of the unused syndromes.

33. The method of claim 31, wherein the same logical value is a logical 1, and wherein each bit in the substituted check bits in each code word equals an inverse logical value of each bit in one of the unused syndromes.

34. A data processing system comprising:

a storage array comprising at least one mass storage device;

a host computer system coupled to provide data to the storage array; and

error correction/detection logic configured to generate check bits for the data being provided to the storage array;

wherein the host computer system is configured to test the error correction/detection logic by providing each of a set of n data bit combinations to the error detection/correction logic, wherein each data bit combination has n bits, wherein each possible value of each data bit is present in at least one of the n data bit combinations, wherein the set of n data bit combinations provided to the error detection/correction logic is a subset of a set of all data bit combinations that it is possible to create using n bits;

wherein in response to being provided with the set of n data bit combinations, the error detection/correction logic is configured to generate a set of check bits for each of the n data bit combinations; and

wherein the host computer system is configured to compare the set of check bits generated by the error correction/detection logic with a known correct set of check bits for each of the n data bit combinations.

35. A method of testing error detection/correction logic, the method comprising:

providing a subset of possible data bit combinations of n data bits to the error detection/correction logic, wherein the subset comprises n data bit combinations, wherein each possible value of each data bit is present in at least one of the n data bit combinations in the subset;

verifying the error detection/correction logic by comparing a set of check bits generated by the error detection/correction logic for each of the n data bit combinations in the subset with a set of known correct check bits;

providing a first set of m+1 test code words to the error detection/correction logic, wherein a first test code word is a correct test code word and where each other test code word in the set of m+1 test code words comprises a single-bit error, wherein each test code word having a single-bit error has the single-bit error at a different bit position than each other test code word that has a single-bit error;

wherein said verifying further comprises comparing a first output of the error detection/correction logic generated in response to said providing a first set of m+1 test code words with a first known correct output for each of the m+1 test code words;

providing a second set of test code words to the error detection/correction logic, wherein each test code word in the second set comprises an error introduced by substituting check bits corresponding to an unused syndrome for a correct set of check bits within a correct code word, wherein each test code word in the second set comprises substituted check bits corresponding to a different unused syndrome than each other test code word in the second set of test code words;

wherein said verifying further comprises comparing a second output of the error detection/correction logic generated in response to said providing a second set of test code words with a second known correct output for each of the m+1 test code words; and

in response to said verifying, indicating whether the error detection/correction logic is operating properly.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.